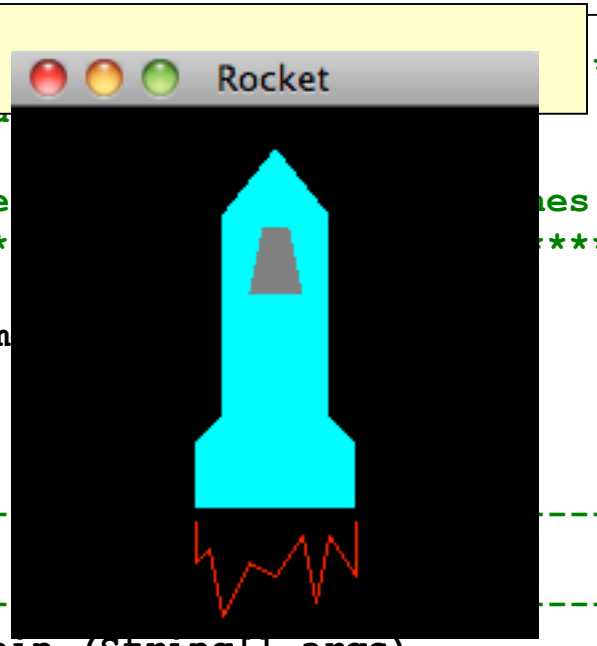
Polygons and Polylines

- Arrays can be helpful in graphics processing
- For example, they can be used to store a list of coordinates
- A *polygon* is a multisided, closed shape
- A *polyline* is similar to a polygon except that its endpoints do not meet, and it cannot be filled
- See `Rocket.java`
- See `RocketPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Rocket.java       Author: Lewis/Loftus
//
//  Demonstrates the use of polygons and polylines.
//********************************************************************

import javax.swing.JFrame;

public class Rocket
{
   //-----------------------------------------------------------------
   //  Creates the main frame of the program.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Rocket");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      RocketPanel panel = new RocketPanel();

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//********************                                    ********************
//  Rocket.java       Au
//
//  Demonstrates the use                                  es.
//********************                                    ********************

import javax.swing.JFram

public class Rocket
{
   //-----------------                                    ---------------------
   //  Creates the main
   //-----------------                                    ---------------------
   public static void main (String[] args)
   {
      JFrame frame = new JFrame("Rocket");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      RocketPanel panel = new RocketPanel();

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  RocketPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of polygons and polylines.
//********************************************************************

import javax.swing.JPanel;
import java.awt.*;

public class RocketPanel extends JPanel
{
   private int[] xRocket = {100, 120, 120, 130, 130, 70, 70, 80, 80};
   private int[] yRocket = {15, 40, 115, 125, 150, 150, 125, 115, 40};

   private int[] xWindow = {95, 105, 110, 90};
   private int[] yWindow = {45, 45, 70, 70};

   private int[] xFlame = {70, 70, 75, 80, 90, 100, 110, 115, 120,
                           130, 130};
   private int[] yFlame = {155, 170, 165, 190, 170, 175, 160, 185,
                           160, 175, 155};
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
   //-----------------------------------------------------------------
   //  Constructor: Sets up the basic characteristics of this panel.
   //-----------------------------------------------------------------
   public RocketPanel()
   {
      setBackground(Color.black);
      setPreferredSize(new Dimension(200, 200));
   }

   //-----------------------------------------------------------------
   //  Draws a rocket using polygons and polylines.
   //-----------------------------------------------------------------
   public void paintComponent(Graphics page)
   {
      super.paintComponent(page);

      page.setColor(Color.cyan);
      page.fillPolygon(xRocket, yRocket, xRocket.length);

      page.setColor(Color.gray);
      page.fillPolygon(xWindow, yWindow, xWindow.length);

      page.setColor(Color.red);
      page.drawPolyline(xFlame, yFlame, xFlame.length);
   }
}
```

Pearson Education, Inc.

The Polygon Class

- The `Polygon` class can also be used to define and draw a polygon
- It is part of the `java.awt` package
- Versions of the overloaded `drawPolygon` and `fillPolygon` methods take a single `Polygon` object as a parameter instead of arrays of coordinates

Copyright © 2014 Pearson Education, Inc.

Mouse Events

Events related to the mouse are separated into *mouse events* and *mouse motion events*

Mouse Events:

mouse pressed	the mouse button is pressed down
mouse released	the mouse button is released
mouse clicked	the mouse button is pressed down and released without moving the mouse in between
mouse entered	the mouse pointer is moved onto (over) a component
mouse exited	the mouse pointer is moved off of a component

Mouse Events

- Mouse motion events:

mouse moved	the mouse is moved
mouse dragged	the mouse is moved while the mouse button is pressed down

- Listeners for mouse events are created using the `MouseListener` and `MouseMotionListener` interfaces
- A `MouseEvent` object is passed to the appropriate method when a mouse event occurs

Copyright © 2014 Pearson Education, Inc.

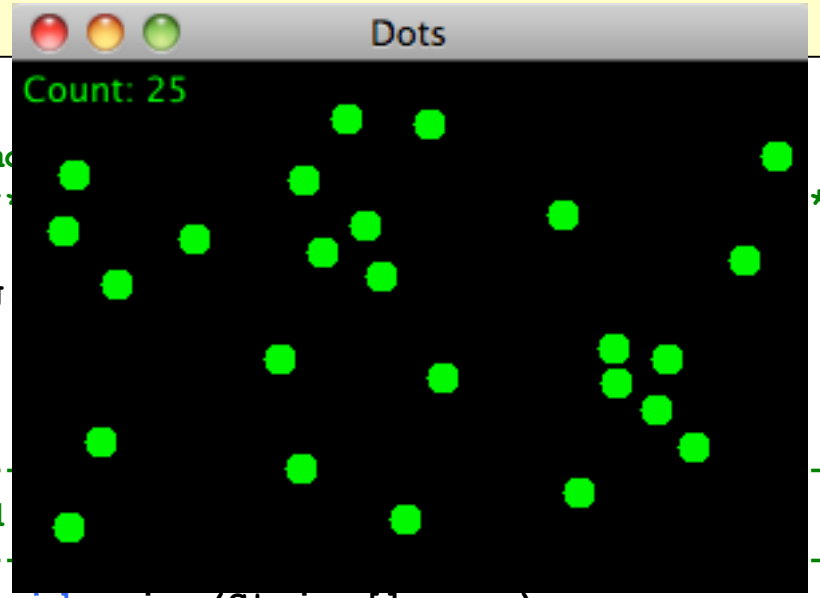
Mouse Events

- For a given program, we may only care about one or two mouse events
- To satisfy the implementation of a listener interface, empty methods must be provided for unused events
- See `Dots.java`
- See `DotsPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Dots.java       Author: Lewis/Loftus
//
//  Demonstrates mouse events.
//********************************************************************

import javax.swing.JFrame;

public class Dots
{
   //-----------------------------------------------------------------
   //  Creates and displays the application frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Dots");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new DotsPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//**************                                          **************
//  Dots.java
//
//  Demonstrates mo                                      ***************

import javax.swing

public class Dots
{
   //--------------                                      ---------------
   //  Creates and                                       ---------------
   public static void main (String[] args)
   {
      JFrame frame = new JFrame("Dots");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new DotsPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  DotsPanel.java       Author: Lewis/Loftus
//
//  Represents the primary panel for the Dots program.
//********************************************************************

import java.util.ArrayList;
import javax.swing.JPanel;
import java.awt.*;
import java.awt.event.*;

public class DotsPanel extends JPanel
{
   private final int SIZE = 6;  // radius of each dot

   private ArrayList<Point> pointList;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Constructor: Sets up this panel to listen for mouse events.
//-----------------------------------------------------------------
public DotsPanel()
{
   pointList = new ArrayList<Point>();

   addMouseListener(new DotsListener());
   setBackground(Color.black);
   setPreferredSize(new Dimension(300, 200));
}

//-----------------------------------------------------------------
//  Draws all of the dots stored in the list.
//-----------------------------------------------------------------
public void paintComponent(Graphics page)
{
   super.paintComponent(page);
   page.setColor(Color.green);

   for (Point spot : pointList)
      page.fillOval(spot.x-SIZE, spot.y-SIZE, SIZE*2, SIZE*2);

   page.drawString("Count: " + pointList.size(), 5, 15);
}
```

continue

Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the listener for mouse events.
   //*****************************************************************
   private class DotsListener implements MouseListener
   {
      //--------------------------------------------------------------
      //  Adds the current point to the list of points and redraws
      //  the panel whenever the mouse button is pressed.
      //--------------------------------------------------------------
      public void mousePressed(MouseEvent event)
      {
         pointList.add(event.getPoint());
         repaint();
      }

      //--------------------------------------------------------------
      //  Provide empty definitions for unused event methods.
      //--------------------------------------------------------------
      public void mouseClicked(MouseEvent event) {}
      public void mouseReleased(MouseEvent event) {}
      public void mouseEntered(MouseEvent event) {}
      public void mouseExited(MouseEvent event) {}
   }
}
```

Copyright © 2014 Pearson Education, Inc.

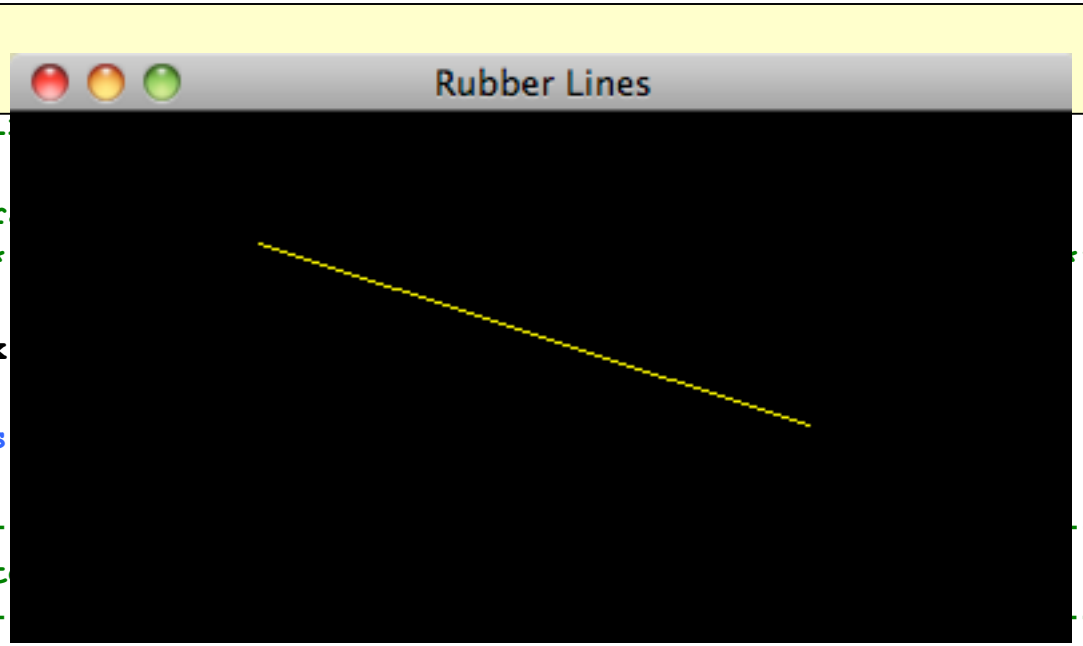
Mouse Events

- *Rubberbanding* is the visual effect in which a shape is "stretched" as it is drawn using the mouse
- The following example continually redraws a line as the mouse is dragged
- See `RubberLines.java`
- See `RubberLinesPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  RubberLines.java       Author: Lewis/Loftus
//
//  Demonstrates mouse events and rubberbanding.
//********************************************************************

import javax.swing.JFrame;

public class RubberLines
{
   //-----------------------------------------------------------------
   //  Creates and displays the application frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Rubber Lines");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new RubberLinesPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//********
//  RubberLi
//
//  Demonstr
//**********

import javax

public class
{
   //-------
   //  Creat
   //-------
   public static void main (String[] args)
   {
      JFrame frame = new JFrame("Rubber Lines");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new RubberLinesPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  RubberLinesPanel.java       Author: Lewis/Loftus
//
//  Represents the primary drawing panel for the RubberLines program.
//********************************************************************

import javax.swing.JPanel;
import java.awt.*;
import java.awt.event.*;

public class RubberLinesPanel extends JPanel
{
   private Point point1 = null, point2 = null;

   //-----------------------------------------------------------------
   //  Constructor: Sets up this panel to listen for mouse events.
   //-----------------------------------------------------------------
   public RubberLinesPanel()
   {
      LineListener listener = new LineListener();
      addMouseListener(listener);
      addMouseMotionListener(listener);

      setBackground(Color.black);
      setPreferredSize(new Dimension(400, 200));
   }
```

continue

Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Draws the current line from the initial mouse-pressed point to
//  the current position of the mouse.
//-----------------------------------------------------------------
public void paintComponent(Graphics page)
{
   super.paintComponent(page);
   page.setColor(Color.yellow);
   if (point1 != null && point2 != null)
      page.drawLine(point1.x, point1.y, point2.x, point2.y);
}

//*****************************************************************
//  Represents the listener for all mouse events.
//*****************************************************************
private class LineListener implements MouseListener, MouseMotionListener
{
   //--------------------------------------------------------------
   //  Captures the initial position at which the mouse button is
   //  pressed.
   //--------------------------------------------------------------
   public void mousePressed(MouseEvent event)
   {
      point1 = event.getPoint();
   }
```

continue

on Education, Inc.

continue

```
      //-----------------------------------------------------------
      //  Gets the current position of the mouse as it is dragged and
      //  redraws the line to create the rubberband effect.
      //-----------------------------------------------------------
      public void mouseDragged(MouseEvent event)
      {
         point2 = event.getPoint();
         repaint();
      }

      //-----------------------------------------------------------
      //  Provide empty definitions for unused event methods.
      //-----------------------------------------------------------
      public void mouseClicked(MouseEvent event) {}
      public void mouseReleased(MouseEvent event) {}
      public void mouseEntered(MouseEvent event) {}
      public void mouseExited(MouseEvent event) {}
      public void mouseMoved(MouseEvent event) {}
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Key Events

- A *key event* is generated when the user types on the keyboard

key pressed	a key on the keyboard is pressed down
key released	a key on the keyboard is released
key typed	a key on the keyboard is pressed down and released

- Listeners for key events are created by implementing the `KeyListener` interface
- A `KeyEvent` object is passed to the appropriate method when a key event occurs

Copyright © 2014 Pearson Education, Inc.

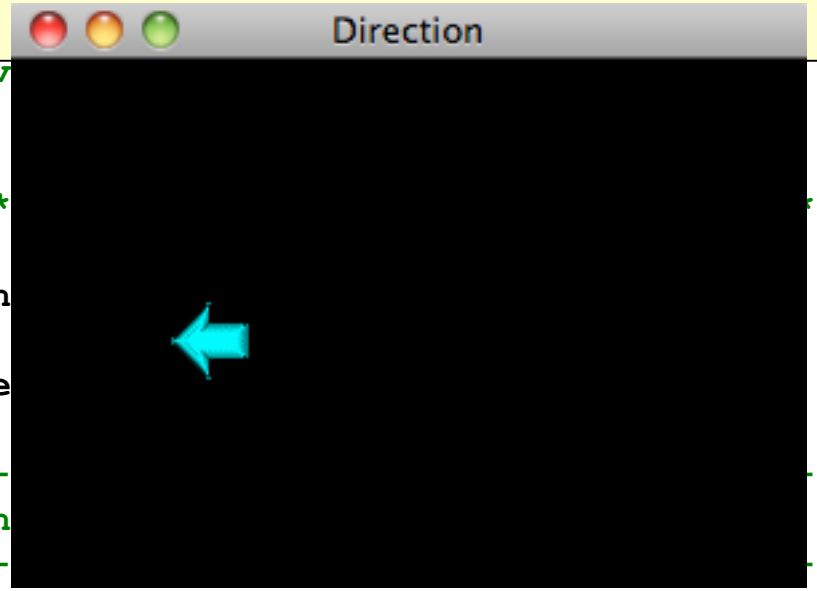
Key Events

- The component that generates a key event is the one that has the current *keyboard focus*
- Constants in the `KeyEvent` class can be used to determine which key was pressed
- The following example "moves" an image of an arrow as the user types the keyboard arrow keys
- See `Direction.java`
- See `DirectionPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Direction.java       Author: Lewis/Loftus
//
//  Demonstrates key events.
//********************************************************************

import javax.swing.JFrame;

public class Direction
{
   //-----------------------------------------------------------------
   //  Creates and displays the application frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Direction");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new DirectionPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//*************
//  Direction.jav
//
//  Demonstrates
//***************

import javax.swin

public class Dire
{
   //------------
   //  Creates an
   //------------
   public static void main (String[] args)
   {
      JFrame frame = new JFrame("Direction");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new DirectionPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//*******************************************************************
//  DirectionPanel.java       Author: Lewis/Loftus
//
//  Represents the primary display panel for the Direction program.
//*******************************************************************

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class DirectionPanel extends JPanel
{
   private final int WIDTH = 300, HEIGHT = 200;
   private final int JUMP = 10;  // increment for image movement

   private final int IMAGE_SIZE = 31;

   private ImageIcon up, down, right, left, currentImage;
   private int x, y;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Constructor: Sets up this panel and loads the images.
//-----------------------------------------------------------------
public DirectionPanel()
{
   addKeyListener(new DirectionListener());

   x = WIDTH / 2;
   y = HEIGHT / 2;

   up = new ImageIcon("arrowUp.gif");
   down = new ImageIcon("arrowDown.gif");
   left = new ImageIcon("arrowLeft.gif");
   right = new ImageIcon("arrowRight.gif");

   currentImage = right;

   setBackground(Color.black);
   setPreferredSize(new Dimension(WIDTH, HEIGHT));
   setFocusable(true);
}
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
   //-----------------------------------------------------------------
   //  Draws the image in the current location.
   //-----------------------------------------------------------------
   public void paintComponent(Graphics page)
   {
      super.paintComponent(page);
      currentImage.paintIcon(this, page, x, y);
   }

   //*****************************************************************
   //  Represents the listener for keyboard activity.
   //*****************************************************************
   private class DirectionListener implements KeyListener
   {
      //--------------------------------------------------------------
      //  Responds to the user pressing arrow keys by adjusting the
      //  image and image location accordingly.
      //--------------------------------------------------------------
      public void keyPressed(KeyEvent event)
      {
         switch (event.getKeyCode())
         {
            case KeyEvent.VK_UP:
               currentImage = up;
               y -= JUMP;
               break;
```

continue

on Education, Inc.

continue

```
            case KeyEvent.VK_DOWN:
               currentImage = down;
               y += JUMP;
               break;
            case KeyEvent.VK_LEFT:
               currentImage = left;
               x -= JUMP;
               break;
            case KeyEvent.VK_RIGHT:
               currentImage = right;
               x += JUMP;
               break;
         }

         repaint();
      }

      //-----------------------------------------------------------
      //  Provide empty definitions for unused event methods.
      //-----------------------------------------------------------
      public void keyTyped(KeyEvent event) {}
      public void keyReleased(KeyEvent event) {}
   }
}
```

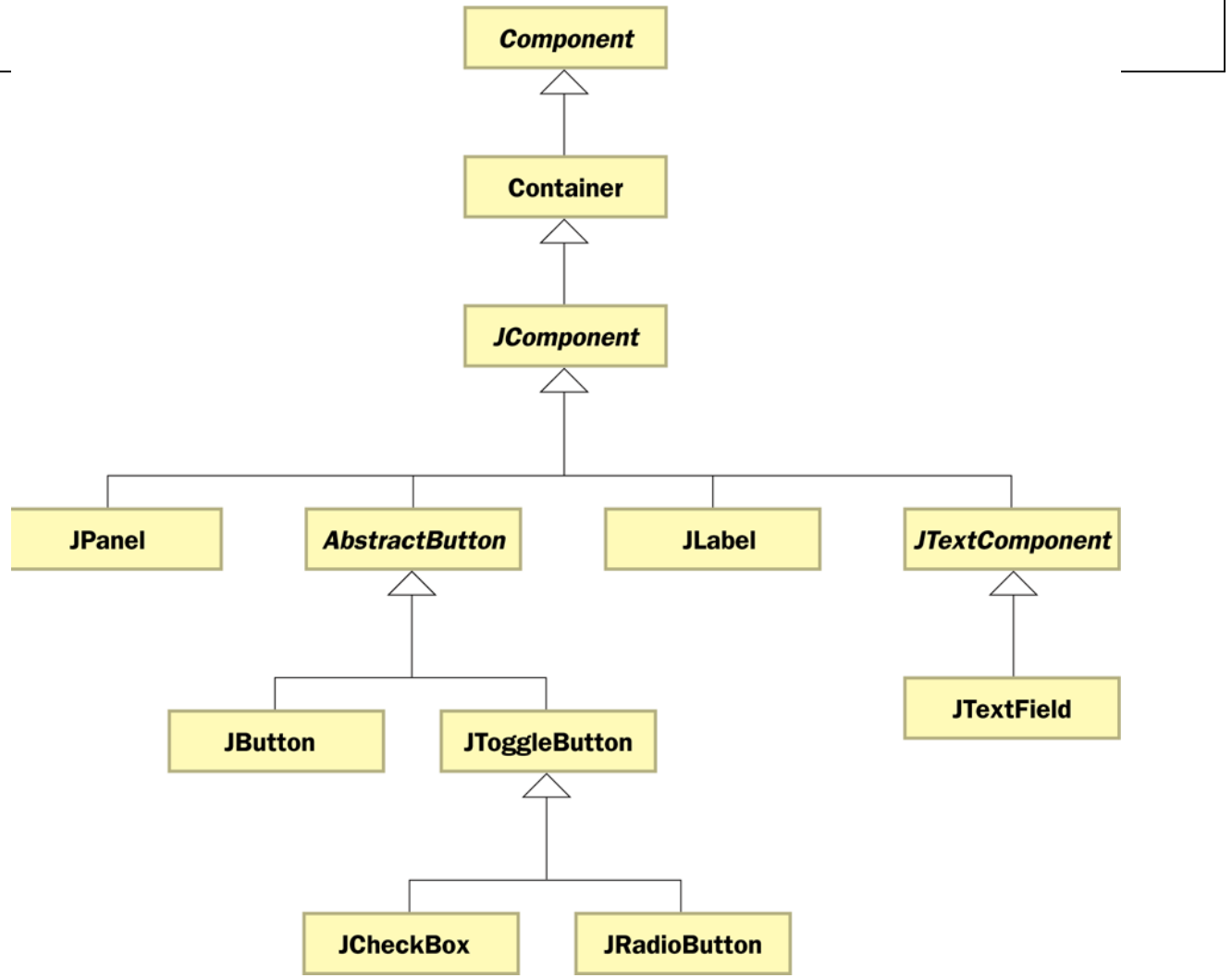
Copyright © 2014 Pearson Education, Inc.

The Component Class Hierarchy

- The Java classes that define GUI components are part of a class hierarchy
- Swing GUI components typically are derived from the `JComponent` class which is derived from the `Container` class which is derived from the `Component` class
- Many Swing components can serve as (limited) containers, because they are derived from the `Container` class
- For example, a `JLabel` object can contain an `ImageIcon`

Copyright © 2014 Pearson Education, Inc.

Partial Component Class Hierarchy



Copyright © 2014 Pearson Education, Inc.

The Component Class Hierarchy

- An applet is another good example of inheritance
- Recall that when we define an applet, we extend the `JApplet` class
- The `JApplet` class already handles all the details about applet creation and execution, including:
  - interaction with a Web browser
  - accepting applet parameters through HTML
  - enforcing security restrictions

Copyright © 2014 Pearson Education, Inc.

The Component Class Hierarchy

- Our applet classes only have to deal with issues that specifically relate to what our particular applet will do
- When we define `paintComponent` method of an applet, we are actually overriding a method defined originally in the `JComponent` class and inherited by the `JApplet` class

Copyright © 2014 Pearson Education, Inc.

Event Adapter Classes

- Inheritance also gives us a alternate technique for creating listener classes
- We've seen that listener classes can be created by implementing a particular interface, such as `MouseListener`
- We can also create a listener class by extending an *event adapter class*
- If a listener interface has more than one method, it has a corresponding adapter class, such as the `MouseAdapter` class

Copyright © 2014 Pearson Education, Inc.

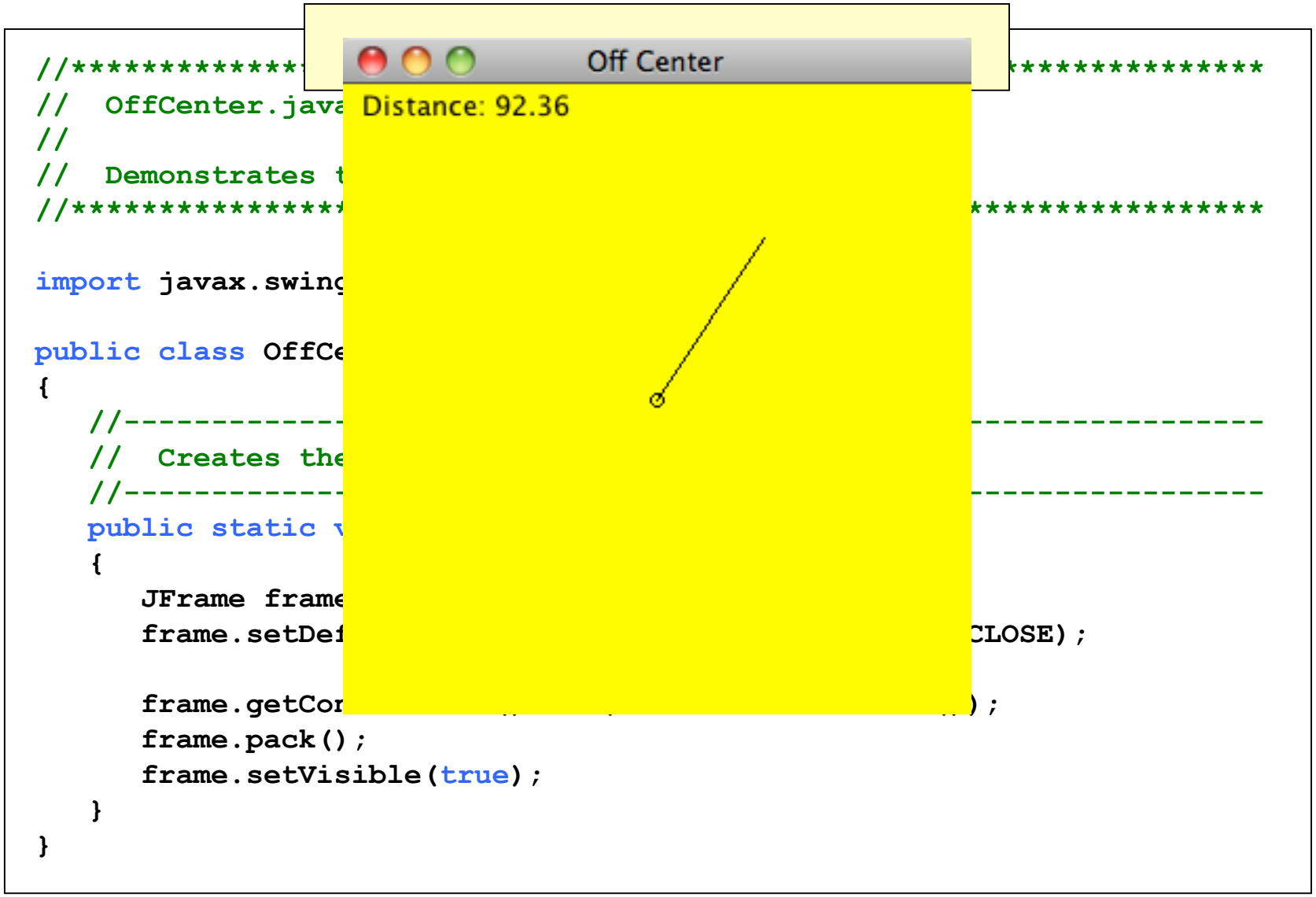
Event Adapter Classes

- Each adapter class implements the corresponding listener, providing empty method definitions
- When you derive a listener class from an adapter class, you only need to override the event methods that pertain to the program
- Empty definitions for unused event methods are automatically provided via inheritance
- See `OffCenter.java`
- See `OffCenterPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  OffCenter.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an event adapter class.
//********************************************************************

import javax.swing.*;

public class OffCenter
{
   //-----------------------------------------------------------------
   //  Creates the main frame of the program.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Off Center");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new OffCenterPanel());
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//*************
//  OffCenter.java
//
//  Demonstrates t
//*****************

import javax.swing

public class OffCe
{
   //-------------
   //  Creates the
   //-------------
   public static v
   {
      JFrame frame
      frame.setDef                                    CLOSE);

      frame.getCon                                    );
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  OffCenterPanel.java       Author: Lewis/Loftus
//
//  Represents the primary drawing panel for the OffCenter program.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import java.text.DecimalFormat;
import javax.swing.*;

public class OffCenterPanel extends JPanel
{
   private final int WIDTH=300, HEIGHT=300;

   private DecimalFormat fmt;
   private Point current;
   private int centerX, centerY;
   private double length;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Constructor: Sets up the panel and necessary data.
//-----------------------------------------------------------------
public OffCenterPanel()
{
   addMouseListener(new OffCenterListener());

   centerX = WIDTH / 2;
   centerY = HEIGHT / 2;

   fmt = new DecimalFormat("0.##");

   setPreferredSize(new Dimension(WIDTH, HEIGHT));
   setBackground(Color.yellow);
}
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Draws a line from the mouse pointer to the center point of
//  the applet and displays the distance.
//-----------------------------------------------------------------
public void paintComponent(Graphics page)
{
   super.paintComponent(page);

   page.setColor(Color.black);
   page.drawOval(centerX-3, centerY-3, 6, 6);

   if (current != null)
   {
      page.drawLine(current.x, current.y, centerX, centerY);
      page.drawString("Distance: " + fmt.format(length), 10, 15);
   }
}
```

continue

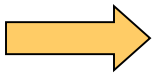
Copyright © 2014 Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the listener for mouse events. Demonstrates the
   //  ability to extend an adaptor class.
   //*****************************************************************
   private class OffCenterListener extends MouseAdapter
   {
      //--------------------------------------------------------------
      //  Computes the distance from the mouse pointer to the center
      //  point of the applet.
      //--------------------------------------------------------------
      public void mouseClicked(MouseEvent event)
      {
         current = event.getPoint();
         length = Math.sqrt(Math.pow((current.x-centerX), 2) +
                            Math.pow((current.y-centerY), 2));
         repaint();
      }
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Outline



Creating Subclasses

Overriding Methods

Class Hierarchies

Visibility

Designing for Inheritance

Inheritance and GUIs

The `Timer` Class

Copyright © 2014 Pearson Education, Inc.

The Timer Class

- The `Timer` class of the `javax.swing` package is a GUI component, but it has no visual representation
- A `Timer` object generates an action event at specified intervals
- Timers can be used to manage any events that are based on a timed interval, such as an animation
- To create the illusion of movement, we use a timer to change the scene after an appropriate delay

Copyright © 2014 Pearson Education, Inc.

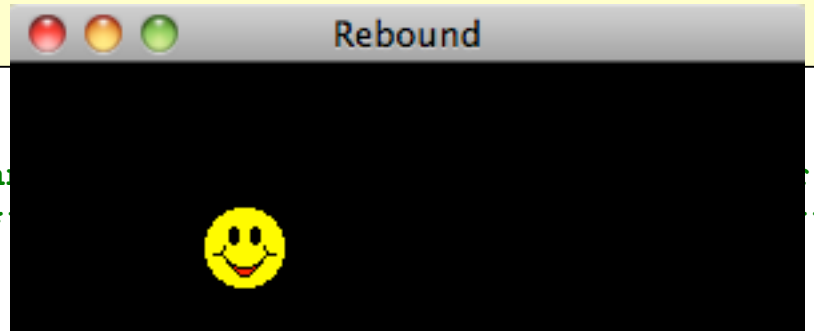
The Timer Class

- The `start` and `stop` methods of the `Timer` class start and stop the timer
- The delay can be set using the `Timer` constructor or using the `setDelay` method
- See `Rebound.java`
- See `ReboundPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Rebound.java       Author: Lewis/Loftus
//
//  Demonstrates an animation and the use of the Timer class.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Rebound
{
   //-----------------------------------------------------------------
   //  Displays the main frame of the program.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Rebound");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new ReboundPanel());
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//********************                                    ********************
//  Rebound.java
//
//  Demonstrates a                                        class.
//********************                                    ********************

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Rebound
{
   //-----------------------------------------------------------------
   //  Displays the main frame of the program.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Rebound");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new ReboundPanel());
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  ReboundPanel.java       Author: Lewis/Loftus
//
//  Represents the primary panel for the Rebound program.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ReboundPanel extends JPanel
{
   private final int WIDTH = 300, HEIGHT = 100;
   private final int DELAY = 20, IMAGE_SIZE = 35;

   private ImageIcon image;
   private Timer timer;
   private int x, y, moveX, moveY;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
    //-----------------------------------------------------------------
    //  Sets up the panel, including the timer for the animation.
    //-----------------------------------------------------------------
    public ReboundPanel()
    {
        timer = new Timer(DELAY, new ReboundListener());
        image = new ImageIcon("happyFace.gif");

        x = 0;
        y = 40;
        moveX = moveY = 3;

        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        setBackground(Color.black);
        timer.start();
    }

    //-----------------------------------------------------------------
    //  Draws the image in the current location.
    //-----------------------------------------------------------------
    public void paintComponent(Graphics page)
    {
        super.paintComponent(page);
        image.paintIcon(this, page, x, y);
    }
```

continue

. Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the action listener for the timer.
   //*****************************************************************
   private class ReboundListener implements ActionListener
   {
      //--------------------------------------------------------------
      //  Updates the position of the image and possibly the direction
      //  of movement whenever the timer fires an action event.
      //--------------------------------------------------------------
      public void actionPerformed(ActionEvent event)
      {
         x += moveX;
         y += moveY;

         if (x <= 0 || x >= WIDTH-IMAGE_SIZE)
            moveX = moveX * -1;

         if (y <= 0 || y >= HEIGHT-IMAGE_SIZE)
            moveY = moveY * -1;

         repaint();
      }
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Event Processing

- Polymorphism plays an important role in the development of a Java graphical user interface
- Consider the following code:

```
JButton button = new JButton();
button.addActionListener(new MyListener());
```

- Note that the `addActionListener` method is accepting a `MyListener` object as a parameter
- In fact, we can pass the `addActionListener` method any object that implements the `ActionListener` interface

Copyright © 2014 Pearson Education, Inc.

Event Processing

- The code for `addActionListener` accepts a parameter of type `ActionListener` (the interface)
- Because of polymorphism, any object that implements that interface is compatible with the parameter reference variable
- The component can call the `actionPerformed` method because of the relationship between the listener class and the interface
- Extending an adapter class to create a listener represents the same situation; the adapter class implements the appropriate interface already

Copyright © 2014 Pearson Education, Inc.

Outline

- Late Binding
- Polymorphism via Inheritance
- Polymorphism via Interfaces
- Sorting
- Searching
- Event Processing Revisited
- File Choosers and Color Choosers
- Sliders

Copyright © 2014 Pearson Education, Inc.

Dialog Boxes

- Recall that a dialog box is a small window that "pops up" to interact with the user for a brief, specific purpose
- We used the `JOptionPane` class in Chapter 6 to create dialog boxes for presenting information, confirming an action, or accepting an input value
- Let's now look at two other classes that let us create specialized dialog boxes

Copyright © 2014 Pearson Education, Inc.

File Choosers

- Situations often arise where we want the user to select a file stored on a disk drive, usually so that its contents can be read and processed
- A *file chooser*, represented by the `JFileChooser` class, simplifies this process
- The user can browse the disk and filter the file types displayed
- See `DisplayFile.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  DisplayFile.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a file chooser and a text area.
//********************************************************************

import java.util.Scanner;
import java.io.*;
import javax.swing.*;

public class DisplayFile
{
   //-----------------------------------------------------------------
   //  Opens a file chooser dialog, reads the selected file and
   //  loads it into a text area.
   //-----------------------------------------------------------------
   public static void main(String[] args) throws IOException
   {
      JFrame frame = new JFrame("Display File");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JTextArea ta = new JTextArea(20, 30);
      JFileChooser chooser = new JFileChooser();

      int status = chooser.showOpenDialog(null);
```

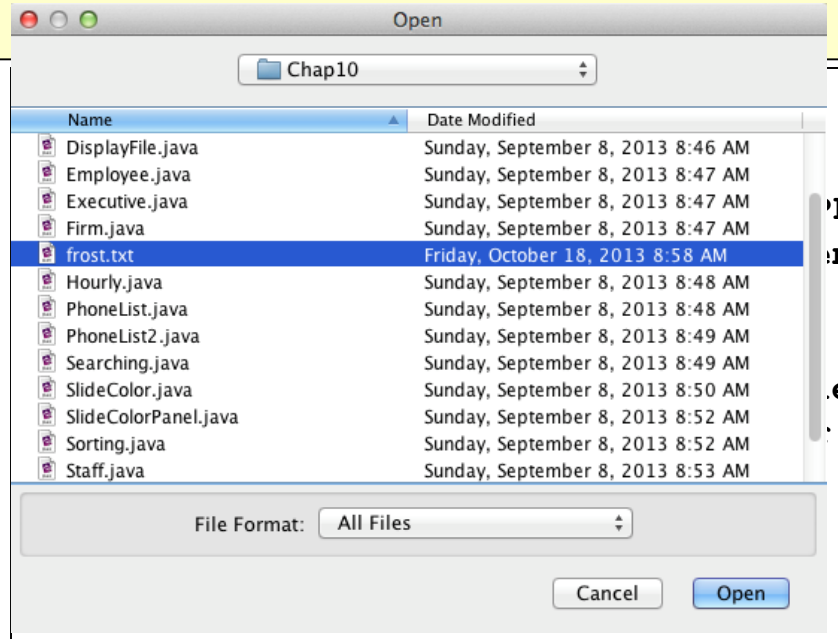
continue

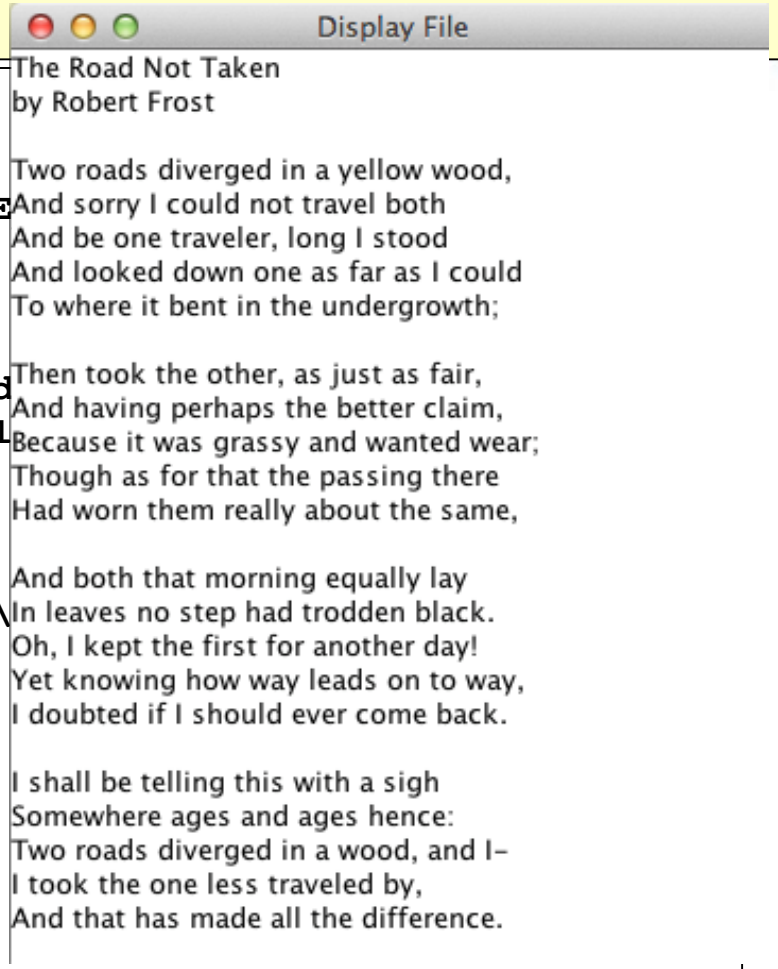
Copyright © 2014 Pearson Education, Inc.

continue

```
      if (status != JFileChooser.APPROVE_OPTION)
         ta.setText("No File Chosen");
      else
      {
         File file = chooser.getSelectedFile();
         Scanner scan = new Scanner(file);

         String info = "";
         while (scan.hasNext())
            info += scan.nextLine() + "\n";

         ta.setText (info);
      }

      frame.getContentPane().add(ta);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.


Open
Chap10
Name
Date Modified
DisplayFile.java Sunday, September 8, 2013 8:46 AM
Employee.java Sunday, September 8, 2013 8:47 AM
Executive.java Sunday, September 8, 2013 8:47 AM
Firm.java Sunday, September 8, 2013 8:47 AM
frost.txt Friday, October 18, 2013 8:58 AM
Hourly.java Sunday, September 8, 2013 8:48 AM
PhoneList.java Sunday, September 8, 2013 8:48 AM
PhoneList2.java Sunday, September 8, 2013 8:49 AM
Searching.java Sunday, September 8, 2013 8:49 AM
SlideColor.java Sunday, September 8, 2013 8:50 AM
SlideColorPanel.java Sunday, September 8, 2013 8:52 AM
Sorting.java Sunday, September 8, 2013 8:52 AM
Staff.java Sunday, September 8, 2013 8:53 AM
File Format:
All Files
Cancel
Open


Display File
The Road Not Taken
by Robert Frost
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;
Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I–
I took the one less traveled by,
And that has made all the difference.

```
PROVE
n");

ected
 (fil

+ "\

            ta.setText (info);
        }

        frame.getContentPane().add(ta);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Color Choosers

- In many situations we want to allow the user to select a color
- A *color chooser*, represented by the `JColorChooser` class, simplifies this process
- The user can choose a color from a palette or specify the color using RGB values
- See `DisplayColor.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  DisplayColor.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a color chooser.
//********************************************************************

import javax.swing.*;
import java.awt.*;

public class DisplayColor
{
   //-----------------------------------------------------------------
   //  Presents a frame with a colored panel, then allows the user
   //  to change the color multiple times using a color chooser.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Display Color");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JPanel colorPanel = new JPanel();
      colorPanel.setBackground(Color.white);
      colorPanel.setPreferredSize(new Dimension(300, 100));
```

continue

Copyright © 2014 Pearson Education, Inc.

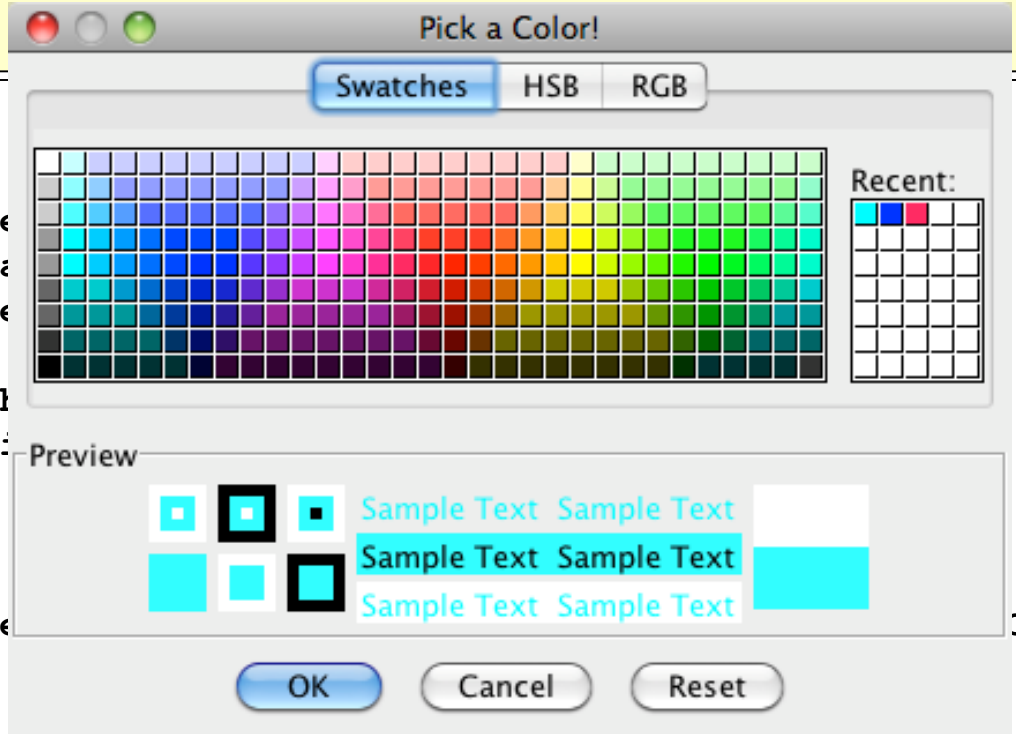
continue

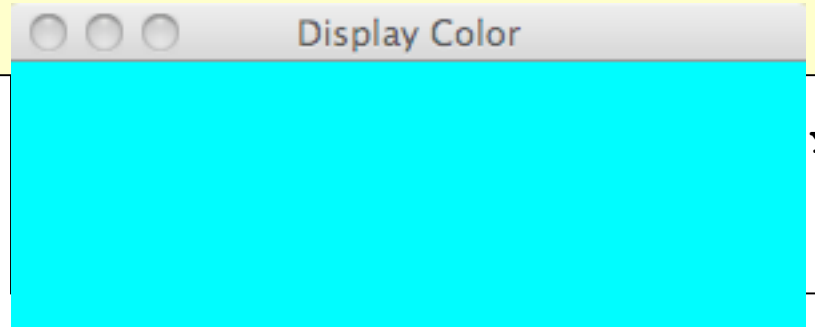
```
        frame.getContentPane().add(colorPanel);
        frame.pack();
        frame.setVisible(true);

        Color shade = Color.white;
        int again;

        do
        {
           shade = JColorChooser.showDialog(frame, "Pick a Color!",
                                            shade);

           colorPanel.setBackground(shade);

           again = JOptionPane.showConfirmDialog (null,
              "Display another color?");
        }
        while (again == JOptionPane.YES_OPTION);
    }
}
```

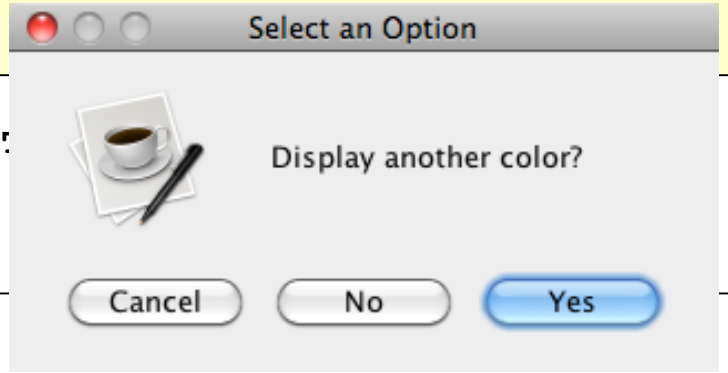
Copyright © 2014 Pearson Education, Inc.

continue

```
        frame.ge
        frame.pa
        frame.se

        Color sh
        int agai

        do
        {
            shade                                    Color!",

            colorPanel.setBackground (shade);

                                                     nfi
                                                     ");

                                                     YES_OP
```







Copyright © 2014 Pearson Education, Inc.

Outline

- Late Binding
- Polymorphism via Inheritance
- Polymorphism via Interfaces
- Sorting
- Searching
- Event Processing Revisited
- File Choosers and Color Choosers
- Sliders

Copyright © 2014 Pearson Education, Inc.

Sliders

- A slider is a GUI component that allows the user to specify a value within a numeric range
- A slider can be oriented vertically or horizontally and can have optional tick marks and labels
- The minimum and maximum values for the slider are set using the `JSlider` constructor
- A slider produces a *change event* when the slider is moved, indicating that the slider and the value it represents has changed

Copyright © 2014 Pearson Education, Inc.

Sliders

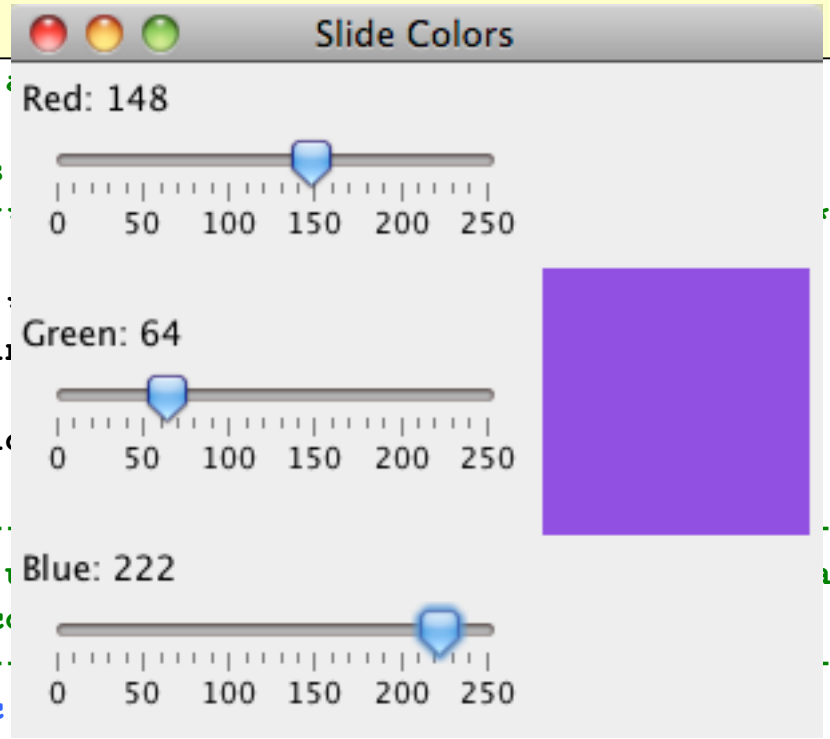
- Let's look at an example that uses three sliders to change values representing the color components of an RGB value

- See `SlideColor.java`
- See `SlideColorPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  SlideColor.java       Authors: Lewis/Loftus
//
//  Demonstrates the use slider components.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class SlideColor
{
   //-----------------------------------------------------------------
   //  Presents up a frame with a control panel and a panel that
   //  changes color as the sliders are adjusted.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Slide Colors");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new SlideColorPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//*************                                          ***************
//  SlideColor.ja
//
//  Demonstrates
//***************                                        ***************

import java.awt.
import javax.swi

public class Sli
{
   //-----------                                        -----------------
   //  Presents                                        a panel that
   //  changes co
   //-----------                                        -----------------
   public static
   {
      JFrame frame = new JFrame("Slide Colors");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new SlideColorPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  SlideColorPanel.java       Authors: Lewis/Loftus
//
//  Represents the slider control panel for the SlideColor program.
//********************************************************************

import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;

public class SlideColorPanel extends JPanel
{
   private JPanel controls, colorPanel;
   private JSlider rSlider, gSlider, bSlider;
   private JLabel rLabel, gLabel, bLabel;

   //-----------------------------------------------------------------
   //  Sets up the sliders and their labels, aligning them along
   //  their left edge using a box layout.
   //-----------------------------------------------------------------
   public SlideColorPanel()
   {
      rSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
      rSlider.setMajorTickSpacing(50);
      rSlider.setMinorTickSpacing(10);
      rSlider.setPaintTicks(true);
      rSlider.setPaintLabels(true);
      rSlider.setAlignmentX(Component.LEFT_ALIGNMENT);
```

continue

Pearson Education, Inc.

continue

```
gSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
gSlider.setMajorTickSpacing(50);
gSlider.setMinorTickSpacing(10);
gSlider.setPaintTicks(true);
gSlider.setPaintLabels(true);
gSlider.setAlignmentX(Component.LEFT_ALIGNMENT);

bSlider = new JSlider(JSlider.HORIZONTAL, 0, 255, 0);
bSlider.setMajorTickSpacing(50);
bSlider.setMinorTickSpacing(10);
bSlider.setPaintTicks(true);
bSlider.setPaintLabels(true);
bSlider.setAlignmentX(Component.LEFT_ALIGNMENT);

SliderListener listener = new SliderListener();
rSlider.addChangeListener(listener);
gSlider.addChangeListener(listener);
bSlider.addChangeListener(listener);

rLabel = new JLabel("Red: 0");
rLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
gLabel = new JLabel("Green: 0");
gLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
bLabel = new JLabel("Blue: 0");
bLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
```

continue

Pearson Education, Inc.

continue

```
        controls = new JPanel();
        BoxLayout layout = new BoxLayout(controls, BoxLayout.Y_AXIS);
        controls.setLayout(layout);
        controls.add(rLabel);
        controls.add(rSlider);
        controls.add(Box.createRigidArea(new Dimension(0, 20)));
        controls.add(gLabel);
        controls.add(gSlider);
        controls.add(Box.createRigidArea(new Dimension(0, 20)));
        controls.add(bLabel);
        controls.add(bSlider);

        colorPanel = new JPanel();
        colorPanel.setPreferredSize(new Dimension(100, 100));
        colorPanel.setBackground(new Color(0, 0, 0));

        add(controls);
        add(colorPanel);
    }
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the listener for all three sliders.
   //*****************************************************************
   private class SliderListener implements ChangeListener
   {
      private int red, green, blue;

      //--------------------------------------------------------------
      //  Gets the value of each slider, then updates the labels and
      //  the color panel.
      //--------------------------------------------------------------
      public void stateChanged(ChangeEvent event)
      {
         red = rSlider.getValue();
         green = gSlider.getValue();
         blue = bSlider.getValue();

         rLabel.setText("Red: " + red);
         gLabel.setText("Green: " + green);
         bLabel.setText("Blue: " + blue);

         colorPanel.setBackground(new Color(red, green, blue));
      }
   }
}
```

Pearson Education, Inc.

Tool Tips

- A *tool tip* provides a short pop-up description when the mouse cursor rests momentarily on a component
- A tool tip is assigned using the `setToolTipText` method of a Swing component

```
JButton button = new JButton("Compute");
button.setToolTipText("Calculate size");
```

Copyright © 2014 Pearson Education, Inc.

Mnemonics

- A *mnemonic* is a keyboard alternative for pushing a button or selecting a menu option
- The mnemonic character should be chosen from the component's label, and is underlined
- The user activates the component by holding down the ALT key and pressing the mnemonic character

```
JButton button = new JButton("Calculate");
button.setMnemonic("C");
```

Copyright © 2014 Pearson Education, Inc.

Disabled Components

- Components can be *disabled* if they should not be used
- A disabled component is "grayed out" and will not respond to user interaction
- The status is set using the `setEnabled` method:

```
JButton button = new JButton("Do It");
button.setEnabled(false);
```

Copyright © 2014 Pearson Education, Inc.

Light Bulb Example

- The right combination of special features such as tool tips and mnemonics can enhance the usefulness of a GUI

- See `LightBulb.java`
- See `LightBulbPanel.java`
- See `LightBulbControls.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  LightBulb.java       Author: Lewis/Loftus
//
//  Demonstrates mnemonics and tool tips.
//********************************************************************

import javax.swing.*;
import java.awt.*;

public class LightBulb
{
   //-----------------------------------------------------------------
   //  Sets up a frame that displays a light bulb image that can be
   //  turned on and off.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Light Bulb");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      LightBulbPanel bulb = new LightBulbPanel();
      LightBulbControls controls = new LightBulbControls(bulb);
```

continue

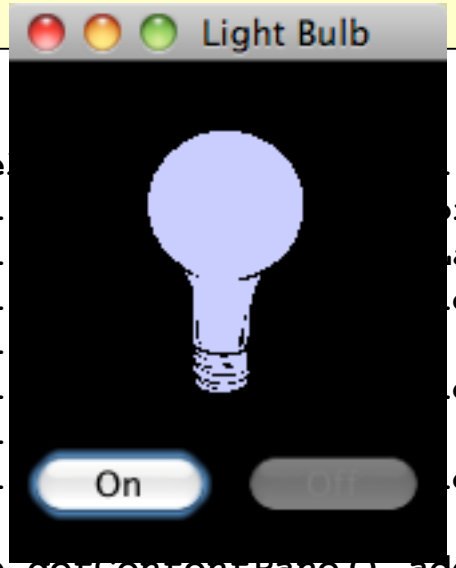
Copyright © 2014 Pearson Education, Inc.

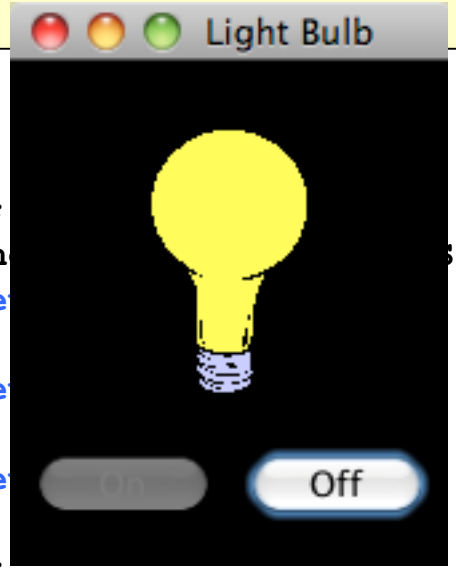
continue

```
        JPanel panel = new JPanel();
        panel.setBackground(Color.black);
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.add(Box.createRigidArea(new Dimension (0, 20)));
        panel.add(bulb);
        panel.add(Box.createRigidArea(new Dimension (0, 10)));
        panel.add(controls);
        panel.add(Box.createRigidArea(new Dimension (0, 10)));

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

continue






```
        JPane                    ();
        panel                    r.black);
        panel                    ayout(pan                    ));
        panel                    dArea (ne                    );
        panel
        panel                    dArea (ne                    );
        panel
        panel                    dArea (ne                    );

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  LightBulbPanel.java       Author: Lewis/Loftus
//
//  Represents the image for the LightBulb program.
//********************************************************************

import javax.swing.*;
import java.awt.*;

public class LightBulbPanel extends JPanel
{
   private boolean on;
   private ImageIcon lightOn, lightOff;
   private JLabel imageLabel;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Constructor: Sets up the images and the initial state.
//-----------------------------------------------------------------
public LightBulbPanel()
{
   lightOn = new ImageIcon("lightBulbOn.gif");
   lightOff = new ImageIcon("lightBulbOff.gif");

   setBackground(Color.black);

   on = true;
   imageLabel = new JLabel(lightOff);
   add(imageLabel);
}
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
   //-----------------------------------------------------------------
   //  Paints the panel using the appropriate image.
   //-----------------------------------------------------------------
   public void paintComponent(Graphics page)
   {
      super.paintComponent(page);

      if (on)
         imageLabel.setIcon(lightOn);
      else
         imageLabel.setIcon(lightOff);
   }

   //-----------------------------------------------------------------
   //  Sets the status of the light bulb.
   //-----------------------------------------------------------------
   public void setOn(boolean lightBulbOn)
   {
      on = lightBulbOn;
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//*******************************************************************
//  LightBulbControls.java       Author: Lewis/Loftus
//
//  Represents the control panel for the LightBulb program.
//*******************************************************************

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class LightBulbControls extends JPanel
{
   private LightBulbPanel bulb;
   private JButton onButton, offButton;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Sets up the lightbulb control panel.
//-----------------------------------------------------------------
public LightBulbControls(LightBulbPanel bulbPanel)
{
   bulb = bulbPanel;

   onButton = new JButton("On");
   onButton.setEnabled(false);
   onButton.setMnemonic('n');
   onButton.setToolTipText("Turn it on!");
   onButton.addActionListener(new OnListener());

   offButton = new JButton("Off");
   offButton.setEnabled(true);
   offButton.setMnemonic('f');
   offButton.setToolTipText("Turn it off!");
   offButton.addActionListener(new OffListener());

   setBackground(Color.black);
   add(onButton);
   add(offButton);
}
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//*****************************************************************
//  Represents the listener for the On button.
//*****************************************************************
private class OnListener implements ActionListener
{
   //--------------------------------------------------------------
   //  Turns the bulb on and repaints the bulb panel.
   //--------------------------------------------------------------
   public void actionPerformed(ActionEvent event)
   {
      bulb.setOn(true);
      onButton.setEnabled(false);
      offButton.setEnabled(true);
      bulb.repaint();
   }
}
```

continue

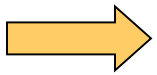
Copyright © 2014 Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the listener for the Off button.
   //*****************************************************************
   private class OffListener implements ActionListener
   {
      //--------------------------------------------------------------
      //  Turns the bulb off and repaints the bulb panel.
      //--------------------------------------------------------------
      public void actionPerformed(ActionEvent event)
      {
         bulb.setOn(false);
         onButton.setEnabled(true);
         offButton.setEnabled(false);
         bulb.repaint();
      }
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Outline



- Exception Handling
- The try-catch Statement
- Exception Classes
- I/O Exceptions
- Tool Tips and Mnemonics
- Combo Boxes
- Scroll Panes and Split Panes

Copyright © 2014 Pearson Education, Inc.

Combo Boxes

- A *combo box* provides a menu from which the user can choose one of several options
- The currently selected option is shown in the combo box
- A combo box shows its options only when the user presses it using the mouse
- Options can be established using an array of strings or using the `addItem` method

Copyright © 2014 Pearson Education, Inc.

The JukeBox Program

- A combo box generates an action event when the user makes a selection from it
- See `JukeBox.java`
- See `JukeBoxControls.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  JukeBox.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a combo box.
//********************************************************************

import javax.swing.*;

public class JukeBox
{
   //-----------------------------------------------------------------
   //  Creates and displays the controls for a juke box.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Java Juke Box");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JukeBoxControls controlPanel = new JukeBoxControls();

      frame.getContentPane().add(controlPanel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//**********
//  JukeBox.ja
//
//  Demonstrates
//**************

import javax.swi

public class JukeBox
{
   //-----------------------------------------------------------------
   //  Creates and displays the controls for a juke box.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Java Juke Box");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JukeBoxControls controlPanel = new JukeBoxControls();

      frame.getContentPane().add(controlPanel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  JukeBoxControls.java       Author: Lewis and Loftus
//
//  Represents the control panel for the juke box.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.AudioClip;
import java.net.URL;

public class JukeBoxControls extends JPanel
{
   private JComboBox musicCombo;
   private JButton stopButton, playButton;
   private AudioClip[] music;
   private AudioClip current;

   //-----------------------------------------------------------------
   //  Sets up the GUI for the juke box.
   //-----------------------------------------------------------------
   public JukeBoxControls()
   {
      URL url1, url2, url3, url4, url5, url6;
      url1 = url2 = url3 = url4 = url5 = url6 = null;
```

continue

Pearson Education, Inc.

continue

```
// Obtain and store the audio clips to play
try
{
   url1 = new URL("file", "localhost", "westernBeat.wav");
   url2 = new URL("file", "localhost", "classical.wav");
   url3 = new URL("file", "localhost", "jeopardy.au");
   url4 = new URL("file", "localhost", "newAgeRythm.wav");
   url5 = new URL("file", "localhost", "eightiesJam.wav");
   url6 = new URL("file", "localhost", "hitchcock.wav");
}
catch (Exception exception) {}

music = new AudioClip[7];
music[0] = null;  // Corresponds to "Make a Selection..."
music[1] = JApplet.newAudioClip(url1);
music[2] = JApplet.newAudioClip(url2);
music[3] = JApplet.newAudioClip(url3);
music[4] = JApplet.newAudioClip(url4);
music[5] = JApplet.newAudioClip(url5);
music[6] = JApplet.newAudioClip(url6);

JLabel titleLabel = new JLabel("Java Juke Box");
titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
        musicCombo = new JComboBox(musicNames);
        musicCombo.setAlignmentX(Component.CENTER_ALIGNMENT);

        //  Set up the buttons
        playButton = new JButton("Play", new ImageIcon("play.gif"));
        playButton.setBackground(Color.white);
        playButton.setMnemonic('p');
        stopButton = new JButton("Stop", new ImageIcon("stop.gif"));
        stopButton.setBackground(Color.white);
        stopButton.setMnemonic('s');

        JPanel buttons = new JPanel();
        buttons.setLayout(new BoxLayout(buttons, BoxLayout.X_AXIS));
        buttons.add(playButton);
        buttons.add(Box.createRigidArea(new Dimension(5,0)));
        buttons.add(stopButton);
        buttons.setBackground(Color.cyan);

        //  Set up this panel
        setPreferredSize(new Dimension(300, 100));
        setBackground(Color.cyan);
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
      musicCombo.addActionListener(new ComboListener());
      stopButton.addActionListener(new ButtonListener());
      playButton.addActionListener(new ButtonListener());

      current = null;
   }

   //*****************************************************************
   //  Represents the action listener for the combo box.
   //*****************************************************************
   private class ComboListener implements ActionListener
   {
      //--------------------------------------------------------------
      //  Stops playing the current selection (if any) and resets
      //  the current selection to the one chosen.
      //--------------------------------------------------------------
      public void actionPerformed(ActionEvent event)
      {
         if (current != null)
            current.stop();

         current = music[musicCombo.getSelectedIndex()];
      }
    }
```

continue

Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the action listener for both control buttons.
   //*****************************************************************
   private class ButtonListener implements ActionListener
   {
      //--------------------------------------------------------------
      //  Stops the current selection (if any) in either case. If
      //  the play button was pressed, start playing it again.
      //--------------------------------------------------------------
      public void actionPerformed(ActionEvent event)
      {
         if (current != null)
            current.stop();

         if (event.getSource() == playButton)
            if (current != null)
               current.play();
      }
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Outline

Exception Handling

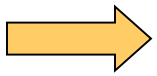
The try-catch Statement

Exception Classes

I/O Exceptions

Tool Tips and Mnemonics

Combo Boxes



Scroll Panes and Split Panes

Copyright © 2014 Pearson Education, Inc.

Scroll Panes

- A *scroll pane* is useful for images or information too large to fit in a reasonably-sized area
- A scroll pane offers a limited view of the component it contains
- It provides vertical and/or horizontal scroll bars that allow the user to scroll to other areas of the component
- No event listener is needed for a scroll pane
- See `TransitMap.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  TransitMap.java       Authors: Lewis/Loftus
//
//  Demonstrates the use a scroll pane.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class TransitMap
{
   //-----------------------------------------------------------------
   //  Presents a frame containing a scroll pane used to view a large
   //  map of the New York subway system.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
       JFrame frame = new JFrame("New York Transit Map");
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        ImageIcon image = new ImageIcon("newyork.jpg");
        JLabel imageLabel = new JLabel(image);

        JScrollPane sp = new JScrollPane(imageLabel);
        sp.setPreferredSize(new Dimension(450, 400));

        frame.getContentPane().add(sp);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

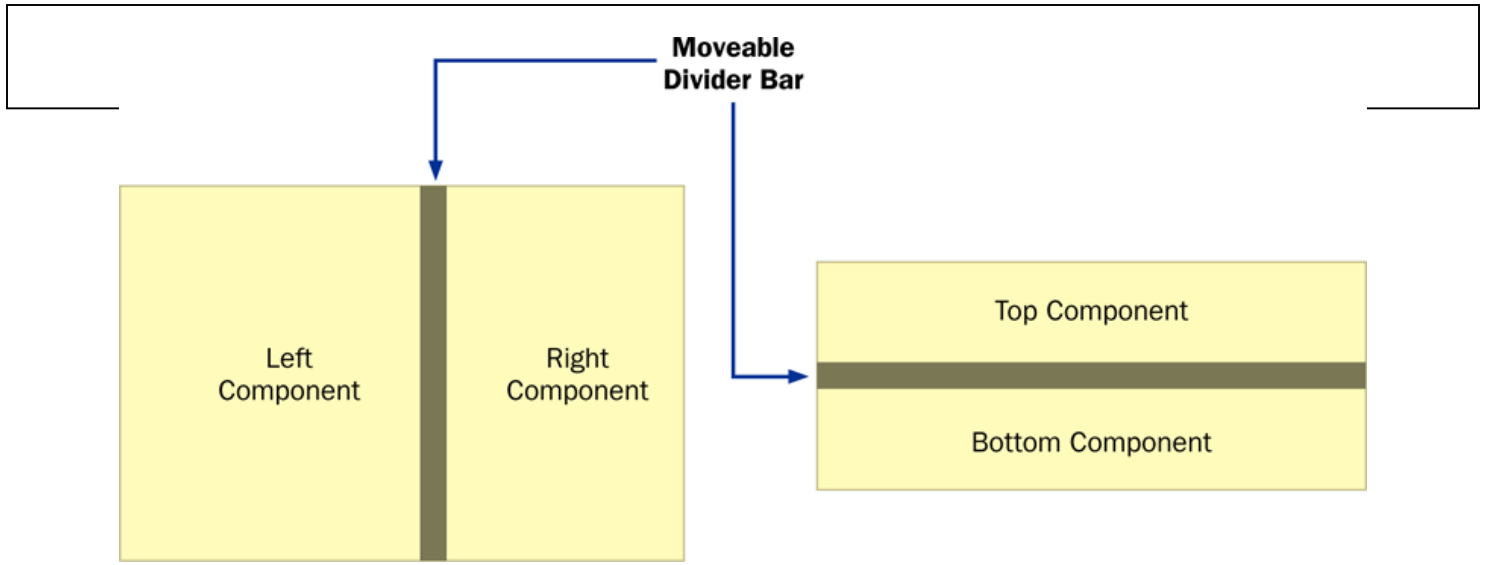
continue

```
        fra

        Ima
        JLa

        JSc
        sp.

        fra
        fra
        fra
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Split Panes

- A split pane is a container that displays two components separated by a moveable divider bar
- The two components can be displayed side by side, or one on top of the other



Copyright © 2014 Pearson Education, Inc.

Split Panes

- The orientation of the split pane is set using the `HORIZONTAL_SPLIT` or `VERTICAL_SPLIT` constants
- The divider bar can be set so that it can be fully expanded with one click of the mouse
- The components can be continuously adjusted as the divider bar is moved, or wait until it stops moving
- Split panes can be nested

Copyright © 2014 Pearson Education, Inc.

Lists

- The Swing `Jlist` class represents a list of items from which the user can choose
- The contents of a `JList` object can be specified using an array of objects
- A `JList` object generates a *list selection event* when the current selection changes
- See `PickImage.java`
- See `ListPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  PickImage.java       Authors: Lewis/Loftus
//
//  Demonstrates the use a split pane and a list.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class PickImage
{
   //-----------------------------------------------------------------
   //  Creates and displays a frame containing a split pane. The
   //  user selects an image name from the list to be displayed.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Pick Image");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
        JLabel imageLabel = new JLabel();
        JPanel imagePanel = new JPanel();
        imagePanel.add(imageLabel);
        imagePanel.setBackground(Color.white);

        ListPanel imageList = new ListPanel(imageLabel);

        JSplitPane sp = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                                       imageList, imagePanel);

        sp.setOneTouchExpandable(true);

        frame.getContentPane().add(sp);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

continue

```
        JLabel i
        JPanel i
        imagePar
        imagePar

        ListPane

        JSplitPa                                          AL_SPLIT,
                                                          el);


        sp.setOr

        frame.ge
        frame.pa
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  ListPanel.java       Authors: Lewis/Loftus
//
//  Represents the list of images for the PickImage program.
//********************************************************************

import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;

public class ListPanel extends JPanel
{
   private JLabel label;
   private JList list;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Loads the list of image names into the list.
//-----------------------------------------------------------------
public ListPanel(JLabel imageLabel)
{
   label = imageLabel;

   String[] fileNames = { "circuit.gif",
                          "duke.gif",
                          "hammock.gif",
                          "justin.jpg",
                          "kayla.jpg",
                          "tiger.jpg",
                          "toucan.gif",
                          "worldmap.gif" };

   list = new JList(fileNames);
   list.addListSelectionListener(new ListListener());
   list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

   add(list);
   setBackground(Color.white);
}
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
   //*****************************************************************
   //  Represents the listener for the list of images.
   //*****************************************************************
   private class ListListener implements ListSelectionListener
   {
      public void valueChanged(ListSelectionEvent event)
      {
         if (list.isSelectionEmpty())
            label.setIcon(null);
         else
         {
            String fileName = (String)list.getSelectedValue();
            ImageIcon image = new ImageIcon(fileName);
            label.setIcon(image);
         }
      }
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Lists

- A `JList` object can be set so that multiple items can be selected at the same time
- The *list selection mode* can be one of three options:
  - single selection – only one item can be selected at a time
  - single interval selection – multiple, contiguous items can be selected at a time
  - multiple interval selection – any combination of items can be selected
- The list selection mode is defined by a `ListSelectionModel` object

Copyright © 2014 Pearson Education, Inc.